Incremento de prestaciones en el acceso en Grid de datos

José María Pérez Menor, Félix García Carballeira, Jesús Carretero Pérez,
José Daniel García Sánchez, Alejandro Calderón Mateos
Departamento de Informática. Universidad Carlos III de Madrid
Av. Universidad, 30 Leganés Madrid
josemaria.perez@uc3m.es

Resumen

El modelo de computación Grid ha evolucionado en los últimos años para proporcionar un entorno de computación de altas prestaciones en redes de área amplia. Sin embargo, uno de los mayores problemas se encuentra en las aplicaciones que hacen uso intensivo y masivo de datos. Como solución a los problemas de estas aplicaciones se ha utilizado la replicación. Sin embargo, la replicación clásica adolece de ciertos problemas como la adaptabilidad y la alta latencia del nuevo entorno. Por ello se propone un nuevo algoritmo de replicación y organización de datos que proporciona un acceso de altas prestaciones en un Data Grid.

1. Introducción

Actualmente se puede obtener una gran potencia de cálculo mediante la utilización de clusters y Grids, sin embargo la E/S se ha convertido en el mayor cuello de botella de estos sistemas, al igual que en su día se convirtió en el cuello de botella de la computación de altas prestaciones en los centros de supercomputación. Para algunas aplicaciones Grid que acceden a conjuntos masivos de datos este problema no solo persiste, sino que se ve acrecentado debido a las distancias entre recursos (latencia), al bajo ancho de banda de algunas redes, al gran número de clientes que acceden a los datos y a la gran cantidad de datos que necesitan.

Los sistemas Grid se suelen clasificar en Grid computacionales (Grids) y Grid de datos (Data Grids). En los Grids computacionales prima el aspecto computacional y puede verse como una simple ampliación del concepto de cluster, mientras que en los Grids de datos prima el acceso a grandes cantidades de datos de forma eficiente por lo que no basta con escalar el concepto de cluster [1].

Los Data Grids [2] tratan de suavizar los problemas en el acceso masivo a datos en Grid. Se define un Data Grid como un conjunto de recursos de almacenamiento y elementos de adquisición de datos que tratan de proporcionar a las aplicaciones los datos que éstas necesitan mediante un middleware especializado en la gestión de datos.

El sistema propuesto en este artículo se centra en el aspecto de acercar los datos a los clientes que los utilicen, en la escalabilidad mediante la agrupación física y lógica de recursos, en la reconfigurabilidad y en la autogestión del almacenamiento y del acceso a datos en un Data Grid. La agrupación física trata de explotar la proximidad de localización de los clientes y nodos de almacenamiento, mientras que la agrupación lógica trata de aprovechar la característica *small worlds* [3] presente en los Grid multidisciplinares.

2. Modelo de Grid de datos

El sistema propuesto se centra en el aspecto de almacenamiento y acceso a datos en un Data Grid El objetivo principal de este artículo es proponer mecanismos para optimizar el rendimiento, la escalabilidad y el balance de carga en el acceso y creación de conjuntos de datos entorno a los clientes que los necesiten o puedan necesitar en el futuro en un Data Grid.

Para aumentar el rendimiento en el acceso y la gestión de los datos se utilizarán técnicas de E/S paralela y la agrupación de recursos de forma física (localización) y lógica (contenidos).

Para ello se definirán una serie de entidades que tratan de agrupar los datos y recursos tanto física como lógicamente (ver Figura 1).

En el sistema propuesto un archivo está compuesto de un subarchivo de metadatos y un

archivo de datos. La separación de datos y metadatos se debe al mecanismo de localización tipo peer-to-peer que permite una gran escalabilidad y que permite guardar una clave (subarchivo de metadatos) en cualquier nodo del Data Grid, mientras que el subarchivo de datos permanece cerca de donde sean necesarios.

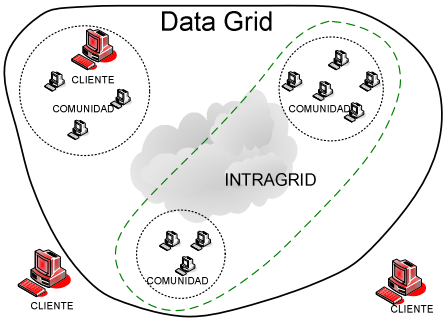

Figura 1. Entidades del sistema Data Grid

Las comunidades de almacenamiento se definen como un conjunto de nodos de almacenamiento cercanos entre sí, cuyo objetivo es proporcionar recursos de almacenamiento a los clientes cercanos, pero que pueden ser accedidos desde cualquier lugar del Grid si es necesario.

Cada comunidad de almacenamiento tiene un nodo central que realizará funciones de localización de datos y de recursos, para lo que será necesario establecer contacto con otras comunidades.

Se define una intragrid como una entidad que agrupa varias comunidades de almacenamiento mediante criterios de intereses comunes. Es decir, se trata de una entidad agrupativa basada en criterios lógicos (contenido de los archivos).

Las intragrids tratan de acercar a los clientes los datos que puedan necesitar sus aplicaciones en el futuro, basándose en el contenido de los datos, reforzándose la relación a medida que se vayan creando nuevos archivos y relajándose cuando una comunidad esté más interesada en otros archivos o conjuntos de datos. Una comunidad se unirá a intragrids ya existentes o abandonará una intragrid a la que pertenezca según cambien los temas de interés de la comunidad. Una comunidad de almacenamiento puede pertenecer a varias intragrids.

2.1. Localización de archivos

El sistema de localización utilizado en el modelo propuesto en este articulo está basado en las redes de tipo *superpeer* [4], donde se forma una red de cobertura *(overlay network)* entre los nodos centrales (*superpeers*), conectándose el resto de nodos a éstos. Sin embargo, a diferencia de los sistemas *superpeer*, el nodo central no tiene por qué conocer la localización de todos los archivos, ya que dentro de la comunidad puede haber cualquier tipo de sistema de localización.

El sistema de localización distribuido propuesto consta de dos pasos. El primero localiza la comunidad de almacenamiento donde se encuentra el subarchivo de metadatos y en el segundo paso se procede a localizar el archivo de metadatos en la comunidad.

La localización de la comunidad se llevará a cabo mediante una red de cobertura (*overlay network*) formada por todas las comunidades. Ya que normalmente los algoritmos para formar este tipo de redes se basan en nodos, más que en la unión de subredes (comunidades) se utilizará el nodo central de cada comunidad como el representante de la misma. El nodo central de una comunidad no tiene por qué saber en qué nodo de se encuentra un archivo, pero sí sabrá si el archivo se encuentra o no en su comunidad.

El segundo paso de la localización utilizará un mecanismo local a la comunidad para localizar el archivo. Como mecanismo de localización se puede utilizar cualquiera de los siguientes:

- **Centralizados:** El nodo central conoce donde se encuentran los datos. Es un típico servicio de directorios.
- **Distribuido con función de localización:** Se utiliza una función tipo *hash* aplicada al nombre del archivo para localizar el nodo donde está almacenado.
- **Distribuido con multidifusión**: Se utilizan mecanismos de broadcast o multidifusión a todos los nodos para preguntar por un archivo, el nodo que almacene el archivo devolverá la respuesta al nodo que preguntó.

2.2. Creación de archivos

La creación de un archivo sigue los siguientes pasos:

1. Localización del nodo que almacenará el subarchivo de datos.
2. Reserva de recursos mediante la creación del subarchivo de datos.
3. Creación del subarchivo de metadatos.

En un Data Grid uno de los problemas principales es la localización de recursos. Si además se desea que ésta sea eficiente desde el punto de vista de quién utilizará en el futuro esos recursos y del estado del sistema, el problema se complica.

Una complicación adicional es la heterogeneidad de la infraestructura, donde un nodo cercano puede no ser adecuado para servir datos a ciertas aplicaciones (a la hora de crear un archivo se pueden exigir unas condiciones mínimas de distancia y rendimiento), para ello se han de tener en cuenta diferentes métricas: de distancia (Δ), de rendimiento de red (N) y de rendimiento del servidor (Π).

El sistema utiliza los siguientes pasos para la selección de un nodo *adecuado* para almacenar los datos:

1. Primero se trata de localizar un nodo con espacio suficiente en la propia comunidad, teniendo en cuenta las características de cada nodo (Δ, N, Π). De esta forma se reforzará la localidad en el acceso entre nodos de una misma comunidad.
2. Si en la comunidad no se encuentra ningún nodo, el nodo central obtendrá una lista de las comunidades que pertenecen a intragrids interesadas en los mismos contenidos que el archivo que se desea crear. Esta lista de comunidades se ordenará de menor a mayor distancia (Δ) y a continuación se procederá a consultar a las comunidades de la lista hasta localizar un nodo que cumpla con los requisitos de espacio, distancia y rendimiento. De esta forma se refuerza la localidad lógica al utilizar recursos de las intragrids.
3. Si todavía no se encuentra un nodo con recursos suficientes o que se adapte a las características indicadas por el cliente se tratará de localizar un nodo del resto de comunidades del Data Grid ordenadas por distancia (Δ).

3. Algoritmo de replicación

Para tratar con el problema de la latencia en sistemas distribuidos se pueden utilizar varios mecanismos. La solución adoptada en la mayoría de Data Grids es la utilización de replicación [2].

Los sistemas la replicación se han utilizado como un mecanismo que permite proveer de datos a las aplicaciones que hacen uso intensivo de éstos, a la vez que proporciona tolerancia a fallos. Sin embargo, debido a la escala del Grid, son necesarios otros mecanismos adicionales a la simple replicación. Por ejemplo, es necesario el movimiento automático y transparente de datos para poder proporcionar datos a las aplicaciones a lo largo de todo el Grid.

Los sistemas de replicación clásicos pueden clasificarse grosso modo en dos tipos: sistemas de replicación estáticos y dinámicos. Los sistemas estáticos determinan el número de réplicas y su localización durante la creación de un archivo, mientras que los sistemas dinámicos permiten que la localización o el número de réplicas cambien acorde con las necesidades de los usuarios o el estado del sistema en cada momento.

En los primeros Data Grids [5] se utilizaban sistemas estáticos, evolucionando estos hacia sistemas dinámicos en los últimos años [6]. En los sistemas dinámicos los problemas son cuándo, dónde y qué replicar. El sistema propuesto se trata de un sistema adaptativo que proporciona todas estas características, además de orientarse al incremento de prestaciones y escalabilidad en el acceso a datos mediante la aplicación de técnicas de E/S paralela, fragmentación y agrupación lógica de contenidos (intragrids).

3.1. Replicación ramificada

El algoritmo propuesto en este artículo se basa en una organización jerárquica de las réplicas, pero en vez de replicar todo un archivo en un sitio como hace el algoritmo de replicación jerárquico clásico, fragmenta una réplica en varios archivos denominados subréplicas. Cada uno de estos fragmentos se creará en una comunidad de almacenamiento cercana a los clientes que deseen acceder a los datos.

A partir de una réplica raíz o archivo original el sistema va creando subréplicas según las necesidades de los clientes, tratando de que estas subréplicas estén lo más cerca posible de los clientes (ver Figura 2).

Cuando se debe crear una nueva réplica, se calcula el número de subréplicas, el rango de

datos a que da soporte cada subréplica y se determinan las comunidades donde se van a crear éstas (subarchivo de metadatos + subarchivos de datos). A continuación, se fragmenta la réplica en el número de subréplicas deseado, teniendo en cuenta que entre dos subréplicas no puede haber solape de datos.

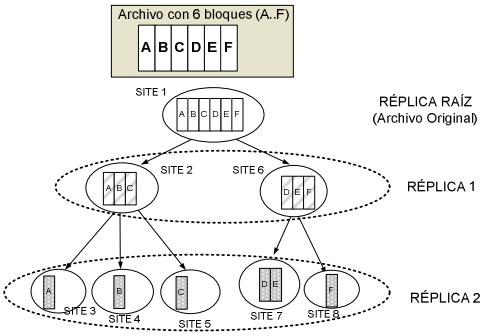

Figura 2. Ejemplo de replicación ramificada

Desde el punto de vista del cliente que abra un archivo, una réplica se define como un conjunto de subréplicas (archivos) que contienen todos los datos de la réplica raíz o archivo original y donde un bloque de datos dado solamente está presente en una de las subréplica (no haya solape).

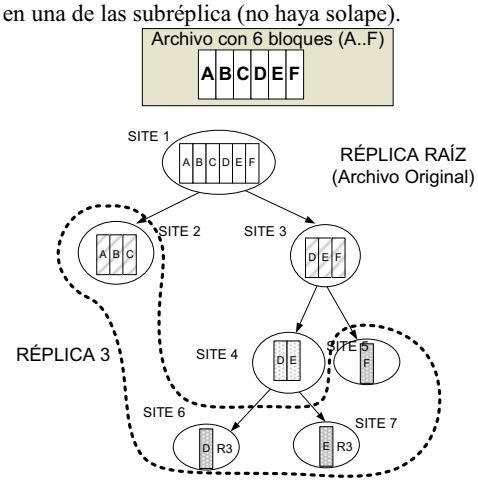

Figura 3 - Ejemplo de selección de réplica

Para saber la carga de cada subréplica en un momento dado, en el subarchivo de metadatos de cada subréplica se almacenará un índice de utilización de un archivo o réplica. Este índice es utilizado en la localización, en la creación y en la destrucción de réplicas.

Este índice de utilización es propio de cada archivo o subréplica y no tiene relación con otros archivos (o subréplicas). Es decir, la utilización de una subréplica no está relacionada con la utilización de sus subréplicas hijas o su réplica padre. Se trata de un valor real entre 0 y 1 que sirve como indicador de lo mucho o poco que es accedida una réplica concreta.

3.2. Localización de réplicas

El problema de localización de réplicas se puede definir de la forma siguiente: Dado un único identificador lógico para un conjunto de datos, determinar la localización física de una o más copias de ese conjunto de datos

Los sistemas de localización de réplicas en los sistemas Data Grid han evolucionado desde directorios de archivos y réplicas, como LDAP, hacia sistemas peer-to-peer [7], pasando por otros mecanismos como filtros Bloom. En la sección 2.3 se presentó el mecanismo para localizar un archivo o réplica raíz (se localiza el subarchivo de metadatos). El sistema de localización de réplicas utiliza este mecanismo distribuido, pero almacena la estructura del árbol de replicación del archivo en el subarchivo de metadatos de la réplica raíz. Esto se hace por razones de eficiencia, ya que siempre podrían hallarse todas las subréplicas siguiendo el árbol de replicación.

Cuando un cliente desee obtener una lista de todas las subréplicas de un archivo (o una sublista, como la de réplicas hoja) accederá a la réplica raíz para obtener la información. Esta información se actualizará con la creación de nuevas subréplicas y ante la desaparición o el fallo de subréplica.

Para mantener la estructura jerárquica, cada subréplica deberá mantener información en sus metadatos de aquellas subréplicas por debajo y por encima de ella en la jerarquía.

3.3. Acceso y creación de réplicas

Desde el punto de vista del cliente, una réplica está formada por un conjunto de subréplicas cercanas que tratan de optimizar el acceso a los datos de un archivo.

La selección de réplicas por parte de un cliente debe tener en cuenta las características del acceso a los datos y el coste de acceso a las subréplicas. Es decir, se utiliza un modelo

económico para la selección de la réplica que produzca menor coste de acceso, pero además para conseguir balance de carga y evitar cuellos de botella se ha de tener en cuenta la carga de las réplicas. Para ello se definen dos parámetros globales: umbral de rendimiento mínimo de una réplica *(MRPT)* y un umbral de utilización máxima de una réplica *(MRUT)*. Para la selección de una réplica (conjunto de subréplicas) de lectura se realizarán los siguientes pasos:

1. Se obtiene el conjunto de todas las réplicas posibles que podrían dar soporte al rango de datos indicado (cada réplica estará formada por un conjunto de subréplicas).
2. Después de determinar el conjunto de réplicas que podría utilizar el cliente, las réplicas que contengan alguna subréplica con un índice de utilización mayor que un umbral de utilización máximo *(I_j > MRUT)* son descartadas con el objetivo de lograr un balance de carga en el acceso al archivo en todo el Data Grid y no saturar solo ciertas réplicas.
3. Para seleccionar la mejor réplica para un cliente se deben tener en cuenta las métricas (Δ, N, Π) para el nodo que almacena el subarchivo de datos. Con este propósito en mente se utilizará una función de rendimiento de acceso *(f_ap)* entre el cliente *(cl)* y el nodo que almacena el subarchivo de datos *(s)*. Para la selección de la mejor réplica se calculará para cada réplica *i* un índice de rendimiento que tiene en cuenta la función de rendimiento de acceso y el índice de utilización del archivo para cada subréplica *j* de la réplica *i*.

$$PI_i = \frac{\sum_{j=1}^{|R_i|} (1 - I_j) \times f\_ap(cl, s)}{|R_i|} \quad (1)$$

4. A continuación, todas las réplicas cuyo índice de rendimiento esté por debajo de un umbral de rendimiento de réplica mínimo *(MRPT)* son descartadas.
5. Finalmente, se selecciona la réplica con mejor índice de rendimiento.

Si después de todo el proceso no hay réplicas que cumplan todas las condiciones se procederá a la creación de una nueva réplica.

La creación de nuevas réplicas se realiza cuando un cliente necesita localizar una réplica de un archivo y no se encuentra ninguna réplica cercana, por lo que el sistema creará una nueva réplica cerca del cliente, o por que las réplicas existentes tienen una alta carga, por lo que se realiza una nueva fragmentación aumentando el grado del paralelismo que se puede alcanzar al acceder al archivo.Se puede decir que un archivo se expande hacia los clientes.

El sistema crece a partir de la réplica raíz formando un árbol que va creciendo y que se va ramificando mediante la fragmentación de las réplicas en varias subréplicas (ramas) según sea necesario, aumentando de esta forma el grado de paralelismo.

El objetivo es crear nuevas subréplicas que permitan que el proceso de búsqueda de réplicas pueda llevarse a cabo en el futuro, para lo que se tratará de fragmentar aquellas réplicas que sean muy utilizadas y estén lejos del cliente. A la hora de seleccionar las réplicas a fragmentar se siguen los siguientes pasos:

1. Se selecciona el conjunto de subréplicas hoja, las subréplicas del nivel más profundo. Para la localización de las réplicas hoja se contactará con el nodo que almacena el subarchivo de metadatos de la réplica raíz que devolverá la lista de subréplicas hoja actual.
2. Para cada subréplica se calcula el índice de rendimiento *(1-I)×f_ap(cl, s)*.
3. Se selecciona de entre las subréplicas aquellas subréplicas con un índice de rendimiento menor que el umbral *MRPT*. Es decir, se fragmentarán aquellas réplicas con *peor rendimiento*.
4. Del conjunto de subréplicas obtenido se seleccionan aquellas subréplicas cuyo índice de utilización sea mayor que el umbral de utilización *MRUT*. Aquellas que son más demandadas serán fragmentadas para obtener un balance de carga, al repartir las peticiones entre la subréplica original y las nuevas subréplicas (subsubréplicas) de ésta, y ya que de esta forma los clientes pueden aumentar el grado de paralelismo.
5. Se debe calcular el número y rango de datos de las subsubréplicas para cada subréplica seleccionada.
6. Cada subsubréplica se crea como un archivo cerca del cliente. Es decir, se creará un archivo en una comunidad de almacenamiento cercana. Para ello el cliente iniciará el proceso de creación de archivos y de localización de recursos. Ya que no se quiere que todas las subréplicas se creen en la misma comunidad,

sino que se repartan entre comunidades cercanas se pedirá al Grid una lista de comunidades cercanas y de entre estas comunidades cercanas se elegirá una de forma aleatoria para crear el archivo (subarchivo de metadatos + archivos de datos) en ésta. En este caso el subarchivo de metadatos y el de datos pueden residir en la misma comunidad ya que los metadatos no son utilizados por el sistema de localización básico del Data Grid.

7. Por último el cliente volverá a iniciar el proceso de búsqueda de réplicas (y también el de creación de réplicas si es necesario) hasta que encuentre una réplica adecuada.

4. Evaluación

Para observar la ganancia del sistema y de los mecanismos propuestos se ha evaluado analíticamente cada una de las características siguientes: E/S paralela, reparto de fragmentos y agrupación lógica (intragrids).
Para la evaluación se partirá de un modelo analítico básico (ver Figura 4).

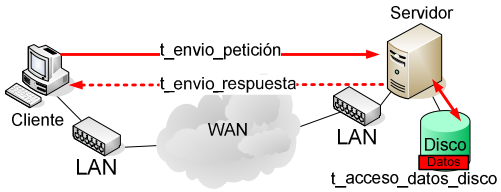

Figura 4. Modelo básico de acceso

Para los valores de los parámetros del disco se han utilizado los del disco Seagate Barracuda ST3160021A 7200RPM.

- 160 GBytes
- t_{seek}: 8,5 ms. $t_{latencia}$: 4,16 ms
- C/H/S: 16383/16/63
- tamaño de sector: 512 bytes
- transferencia media sostenida: 32 a 58 MB/s

Se ha considerado que las redes locales (LAN) son Fast Ethernet, cuyo ancho de banda máximo es 100 Mbits/s = 12,5 Mbytes/s y cuyo ancho de banda efectivo está alrededor de los 8 Mbytes/s. Se ha considerado una latencia para la LAN de 512 microsegundos. Para la red de área amplia (WAN) se ha considerado un ancho de banda de 1 Gigabit/s y un ancho de banda efectivo de 80 Mbytes/s. El valor de la latencia de la WAN se ha dejado como un parámetro de los modelos para observar la influencia de este factor en los diferentes mecanismos propuestos.

4.1. Efectos del paralelismo

Para analizar el efecto del paralelismo en el acceso a datos en un Data Grid y el efecto de la WAN. Se ha calculado el tiempo de acceso para diferentes latencias y número de subréplicas.

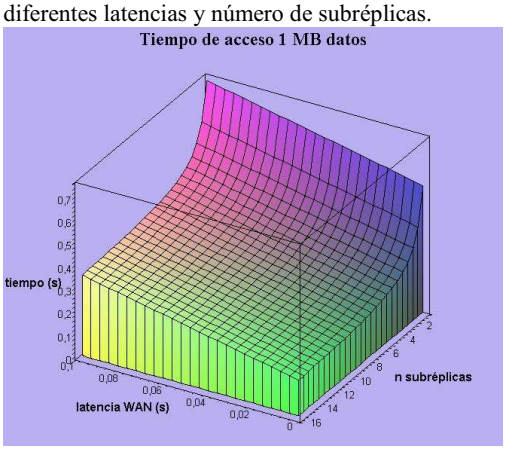

Figura 5. Paralelismo vs. latencia WAN

En la Figura 5 se presenta el acceso a 1 MB de datos, que para los tamaños de archivos manejados en Data Grids puede considerarse un tamaño pequeño. Aun así el paralelismo permite alcanzar un mejor rendimiento en el acceso a datos, aún en redes de área amplia el efecto del paralelismo se deja notar incluso para latencia de 100 ms. La ganancia aportada por el paralelismo se encuentra en la utilización en paralelo de los recursos de comunidades remotas (Disco y LAN de los servidores remotos). Y si se considera que el efecto de la latencia es prácticamente constante, la ganancia será más notable para volúmenes mayores de datos.

4.2. Efectos de la fragmentación

El siguiente mecanismo a analizar es la fragmentación de réplicas. Para ello se comparará en un escenario dado un sistema de réplicas clásico y el propuesto en este trabajo (ver Figura 6 y Figura 7).

En este escenario se tienen 9 comunidades conectadas mediante varias redes (WAN). En un ejemplo del mundo real podría tratarse por ejemplo de una red en Madrid, otra en Barcelona y otra en Valencia, conectadas las tres por otra WAN. En cada una de estas redes habría tres comunidades.

Si un cliente desea acceder a un servidor lejano deberá hacerlo a través de tres WANs.

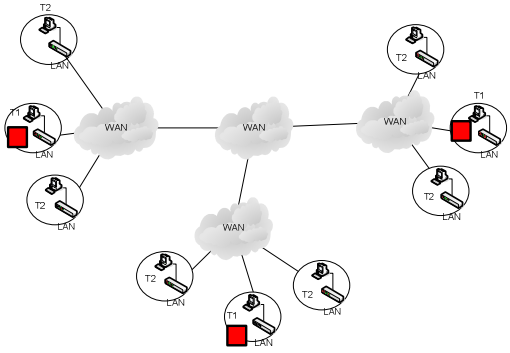

Figura 6. Replicación clásica

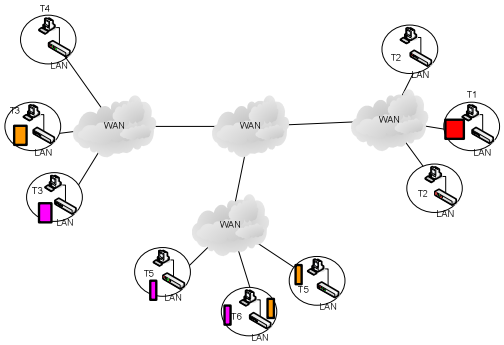

Figura 7. Replicación ramificada

En la Figura 8 y Figura 9 puede verse el tiempo medio de acceso con clientes distribuidos homogéneamente en todo el Grid.

En la Figura 8 se compara el tiempo medio de los clientes en el caso de que haya 2 y 3 réplicas completas (replicación clásica) frente a dos configuraciones de replicación ramificada con 2 réplicas (1 réplica completa y 2 subréplicas de ½), donde con *configuraciones* se hace referencia a distribuciones distintas de las subréplicas.

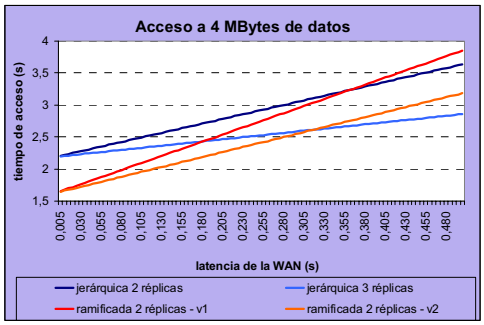

Figura 8. Comparación 2 y 3 réplicas

Como se puede ver, el tiempo de acceso medio con replicación ramificada es incluso mejor que con 3 réplicas completas para latencias de hasta 150 ms (tengase en cuenta que en ese caso que atraviesan hasta 3 WAN, lo que implicaría 450 ms de latencia).

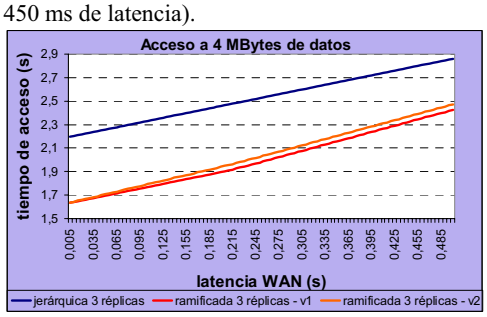

Figura 9. Acceso a tres réplicas

En la Figura 9 se puede ver el resultado en el que se comparán las configuraciones de distribución de las Figuras 6 y 7. Como se puede observar el tiempo medio de acceso es mucho mejor incluso para latencias de WAN de 0,5 segundos.

4.3. Efectos de las intragrids

La entidad intragrid sirve para agrupar lógicamente los datos más utilizados por una serie de comunidades.

Para contrastar el efecto de las intragrids en el acceso a los datos se compararán los modelos anteriores con uno donde varias comunidades están más interesadas en un archivo (es decir los clientes de esas comunidades realizarán un gran número de accesos a esos datos). Para ello sobre la infraestructura utilizada anteriormente se creará

una intragrid formada por cuatro comunidades de las nueve existentes.

La diferencia con los modelos anteriores estriba en que los clientes no están distribuidos homogéneamente por todo el Grid, sino que el número de clientes que acceden desde la intragrid será mayor que desde el resto de comunidades que no pertenecen a la intragrid.

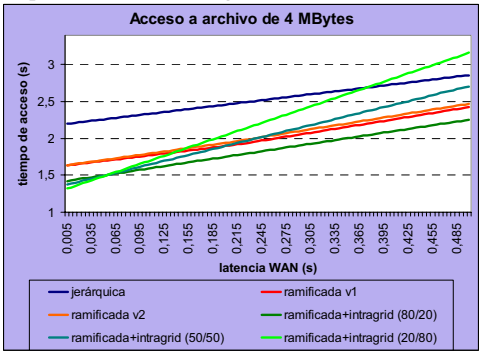

Figura 10. Comparación de acceso con intragrids

En la Figura 11 se pueden comparar los resultados para 3 réplicas (Figura 10) con el efecto de la intragrid para los casos en que el 80%, 50% y 20% de los clientes pertenecen a la intragrid.

Como se puede ver, el efecto de las intragrids permite mejorar el rendimiento. Sin embargo, para latencias de WAN grandes esta ventaja se reduce debido a que aunque el porcentaje de clientes de fuera de la intragrid sea pequeño el efecto de la latencia empieza a ser significativo. Además, la ventaja obtenida se reduce para tamaños de archivo mayores, debido a que el efecto del paralelismo se hace más patente en los clientes lejanos.

5. Conclusiones

En este artículo se han propuesto varios mecanismos que permiten mejorar el acceso a datos en Data Grids. Para ello se ha utilizado una combinación de paralelismo, fragmentación y localidad de los datos mediante un nuevo algoritmo de replicación.

En trabajos futuros se estudiarán los efectos de las escrituras y actualizaciones, que debido a la organización, paralelismo y adaptabilidad del algoritmo de replicación permitirán obtener tiempos de actualización mejores que otros algoritmos.

Referencias

[1] Heinz Stockinger. Distributed Database Management Systems and the Data Grid. 18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies. 2001. San Diego, CA.

[2] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications. 23:187-200, 2001.

[3] Iamnitchi, A., Ripeanu, M., and Foster, I. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. First International Workshop on Peer-to-Peer Systems, Cambridge, Massachusetts, March 2002.

[4] Beverly Yand, Hector Garcia-Molina. Designing a Super-Peer Network. 19th International Conference on Data Engineering. March 05 - 08, 2003. Bangalore, India.

[5] Chervenak, A., Deelman, E., Foster, I., Iamnitchi, A., Kesselman, C., Hoschek, W., Kunszt, P., Ripeanu, M, Schwartzkopf, B., Stockinger, H., Stockinger, K., and Tierney, B. Giggle: A Framework for Constructing Scalable Replica Location Service. Supercomputing'02, November, 2002.

[6] Kavitha Ranganathan, Adriana Iamnitchi and Ian Foster, Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities. Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems, Berlin, May 2002.

[7] A peer-to-Peer Replica Location Service Based on A Distributed Hash Table. Min Cai, ann chervenak, Martin Frank. November 2004. SC2004. The International Conference for High Performance Computing and Communications